Efficient Construction of the Medial Axis for a CAD Model using Parallel Computing

Housheng Zhu[1], Yusheng Liu[2]*, Hongwei Wang[3], Jianjun Zhao[4]*

[1]Information Science and Technology College, Dalian Maritime University, Dalian, 116026
[2]State Key Lab. of CAD&CG, Zhejiang University, Hangzhou, 310027
[3]School of Engineering, University of Portsmouth, Portsmouth, UK, PO1 3DJ
[4]School of Mechanical Science & Engineering, HUST, Wuhan, P.R.China, 430074

Abstract— As a simplified representation of a geometric model, the Medial Axis (MA) has been used in a wide range of engineering applications. While obtaining the true MA of a complicated CAD model is known to be a difficult task, current research is predominantly focused on computing its approximate MA instead. To improve its quality, this work develops a novel and efficient method for obtaining a high-quality MA composed of MA faces for a CAD model. Specifically, an MA point is computed by using a dual-normal-tracing algorithm for each sample point. This algorithm can be implemented through GPU-enabled parallel computing and be executed in an iterative manner until MA points have been found for all sample points. After the iteration is completed, the MA points generated are then converted into the resultant MA by evaluating the topological connectivities of their corresponding sample points. Finally, the resultant MA is converted into MA faces using the information of boundary CAD faces. The proposed method is evaluated by analyzing its complexity and robustness, discussing its applicability and testing its performance in a couple of computational experiments. As shown in the evaluation, this method is easy to implement through exploiting parallel computing and can support effective and high-quality MA generation for a CAD model.

Index Terms—Medial axis, Parallel computing, Mesh model, GPU, Dual-normal-tracing

1 Introduction

The medial axis (MA) was first introduced by Blum [1] as a representation of a geometric object with a reduced number of dimensions. It has been used in many engineering fields such as 3D printing [2], collision detection [3] and surface reconstruction [4] due to its advantageous properties. Different from other representations such as the medial surface and the skeleton, the MA has a strict mathematical definition, i.e. the set of points having more than one closest point on the object's boundary [5].

It is not a trivial task to identify the true MA of a complicated CAD model. Therefore, a lot of research has been conducted to obtain its approximate MA. Those methods, however, have a disadvantage of high time expense or low quality. To the best of the authors' knowledge, most current methods use CPU as the primary processor, resulting in a bottleneck for reducing the time cost of MA generation. The parallel computation by using GPU opens up a new possibility for fast MA computation. However, the approach to dividing the total load for each GPU thread together with the strategy to optimize the division task remains very challenging [6]. Moreover, the resultant MAs of many studies are formed by MA voxels [6-9], MA points [10] or MA seams [11], which cannot facilitate the obtaining of high-quality MA for model analysis, especially when the true MA contains freeform curve surfaces. Meanwhile, the MA faces cannot be extracted from the resultant MA of a CAD model using current methods for analyzing the MA. In this sense, it is highly desirable to develop a method to efficiently generate a high-quality MA composed of MA faces for a CAD model.

Taking account of the considerations mentioned above, a novel algorithm using parallel computing is proposed to

efficiently generate the high-quality MA for a CAD model. Specifically, the MA faces of the resultant MA are generated simultaneously. In the first instance, the input CAD model is discretized into a triangular mesh model using sample points according to the accuracy requirement. As such, the method developed can also be applied to a triangular mesh model or a model which can be converted into a triangular mesh model. Second, the normal directions of boundary points are estimated, which refer to the inner normal directions of boundary points throughout this paper. Then, the corresponding MA point of each sample point is computed in an iterative manner. In each iteration round, MA points for some sample points are obtained until all sample points obtain their MA points in a certain round. Lastly, the MA and its MA faces are generated by extrapolating these MA points and boundary CAD faces as the outputs of the proposed method. The normal directions of all boundary points are all estimated realistically while the generated MA points restrictly follow the definition of the MA using these normal directions. Therefore, the quality of the generated MA is high, as analyzed in the analysis part.

The rest of this paper is organized as follows. In Section 2, the related work is reviewed. After this, an overview of the proposed method is given in Section 3. The pretreatment of the method is explained in Section 4 while Section 5 describes the procedure of computing MA points using the DNT algorithm. The generations of the MA and its MA faces are detailed in Sections 6. The analyses of complexity, robustness, quality and applicability are given in Section 7 while the applications of the method to some examples are detailed in Section 8. Finally, the main conclusions and the directions for future work are given in Section 9.

2 Related Work

Since initially introduced by Blum, the idea of MA has attracted extensive research [12-14]. Previous works in this area can be classified into three categories, namely thinning, tracing and the methods based on Voronoi graph [9, 15-21]. Without parallel computing, some of them can generate a high-quality MA, but require a high expense [9]. Therefore, research on improving computational efficiency for calculating MA is also reviewed in this section. For more detailed information about the MA ideas and concepts, the readers are referred to other published works [22-23].

In the thinning methods, an approximation is usually applied to a model to achieve easy calculation of MA. The calculation precision is controlled by the approximation precision of a model. For example, Lam et al. introduced a method based on a thinning operation [24]. Nackman developed a method which substituted a polygon or polyhedron for the smooth boundaries and on this basis used the MA of the polygon or polyhedron as the MA for the input model [25]. Based on diffusion of the combined waves, Scott et al. proposed a method to determine the symmetric axis of a model, which was a superset of MA [26]. This method is very effective for 0-1 images, but the error of the calculation is considerably greater for color images. Siddiqi et al. proposed a thinning method based on the vector field [27]. In the approach by Vleugels and Overmars, voxelization was recursively performed on a space that contained MA voxels, until the satisfactory resolution was reached [28]. To preserve a model's topology, Borgefors et al. proposed a thinning method for 3D skeletonization, which can be used to calculate both the surface skeleton and the curve skeleton [29]. In this method, the Euclidean distance can be represented by the local distances, and an optimal method is given to calculate the minimum distance between two points [30]. Viswanathan et al. also proposed a thinning method to obtain the MA of a character [31]. In this method, points are deleted from the outer boundary layer by layer until a single pixel-wide skeleton is produced. Stolpner et al. used medial balls to approximate a model [32]. And then, the MA is generated using the centers. However, the execution time in the approximation is not very ideal.

To reduce computational complexity, various strategies have been proposed. In particular, some parallel approaches based on GPU have been proposed to effectively divide computational tasks and exploit computational power of multiple GPUs. For example, GPU has been used in computing exact distance transform for a 3D model [33].

This method is very efficient in generating the MA for a voxel-based model, but it requires expensive cost in converting a B-Rep CAD model into a voxel-based model when computing the MA for a B-Rep CAD model. With such a method, Ma et al. obtained the 3D MA point approximation using the nearest neighbors and the normal field [10]. Jalba et al. improved this work through employing a division-based method as a parallel strategy in addition to the use of GPUs [34]. By using GPU, this method improves the speed of generating the MA dramatically. However, this method is not suitable for CAD models to obtain their MA faces for analyzing the MA. In addition to the GPU-based method, the division-based method was also proved to be an efficient approach. Ramanathan and Gurumoorthy explored how to exract the MA of a model generated by a Boolean union for a 2D model [35]. The boundaries of operand models that are not part of the union were identified and their corresponding MAs were removed. Chang tried to divide a 3D polyhedron into a set of regions using influence shells. Then, the medial surfaces of different regions were computed combined to generate the MA for the entire model [36]. The common characteristic of those division-based methods is that a complicated model is divided into simpler parts for the calculation purpose and then these parts are combined again after the MA calculation process. Sherbrooke et al. built the linear approximate segments of a curve in the direction of the tracing [37]. Their method can be used for conveniently calculating MAs of polyhedrons. Using a piecewise circular boundary conversion, Aichholzer et al. handled models with free-form shapes to efficiently generate the MA [38]. It was claimed that convergence could be ensured with such a method. To speed up the MA generation methods based on Voronoi graph, Meijster et al. separated the grid points of an image into independent rows and columns so as to make it more suitable for parallelization on a shared-memory machine [39]. Hirata devised an efficient algorithm for each row to find the lower envelope of the minima of the distance functions set [40]. Ramamurthy and Farouki developed an incremental method [41]. In this method, a single boundary segment is added to an existing boundary-segment set at each step by modifying the Voronoi regions of the existing boundary segments. The Voronoi region of the new segment is then constructed. After the Voronoi diagram is constructed, the MA is obtained by removing certain edges of the Voronoi diagram that does not belong to the MA and adding certain edges that are outside the Voronoi diagram and belong to the MA. In the authors' earlier work in this area, a voxel-based method was proposed to generate MAs of solid models by reusing the MA of the operand models during the modeling process [6].

Although the MA has beneficial properties, unsmooth boundary with noises can still make its form unstable. To trim the MA, some concepts that are similar to the MA are proposed. For instance, Chaussard et al. proposed a discrete λ-medial axis which proved to be stable under small shape perturbations [42]. The discrete scale axis is proposed to substitute MA as a stable medial representation of 3D models [43]. A representation of medial meshes is proposed as a compact and accurate representation of medial axis transform for 3D models [44]. Li et al also proposed a method using quadratic error minimization to increase the accuracy of the resultant MA [45]. However, the computational complexities of these methods are quite high and the generation of MA faces is not involved.

3 Method Overview

To obtain the MA for a CAD model, in the first instant, the model is discretized into a triangular mesh model with a set of sample points under a certain accuarcy. And then, the space is divided into some grids and the normal directions of boundary points are estimated for the later parallel computing. After that, the medial ball for each sample point is computed. Specifically, according to the definition of the MA, the center of a medial ball is the MA point M of its corresponding sample point P and another boundary point Q on the medial ball is called the ***dual boundary point*** of sample point P. As shown in Fig.1, MA points M_1-M_5 and dual boundary points Q_1-Q_5 of sample points P_1-P_5 are illustrated. The triangle T which contains dual boundary point Q is called the ***dual triangle*** of sample point P. The

radius r of the medial ball of sample point P is called the ***medial radius*** of sample point P. To compute the medial ball for each sample point P, its dual triangle T needs to be searched first. Here, sample point P may have more than one dual triangle and any of them can be considered as the dual triangle T since any of them is sufficient to compute the medial ball for sample point P. Therefore, only one of them needs to be solved as the dual triangle of point P. And then, its MA point M and medial radius r can be determined by solving the equation using the information of sample point P and its dual triangle T. Finally, the MA is generated using the topological relationships of these MA points, and the MA faces are obtained from the MA.

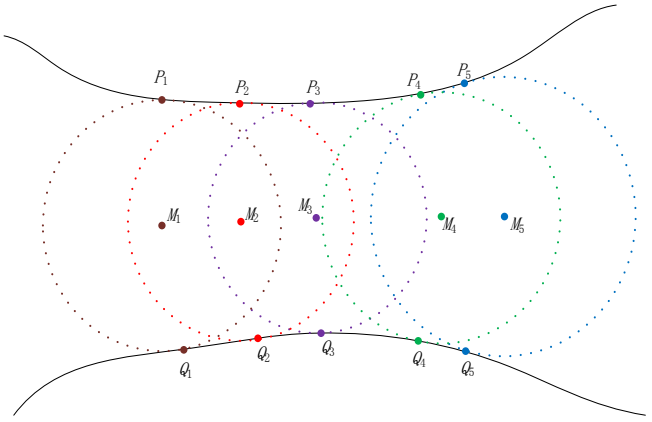


Fig. 1 MA points M_1-M_5 and dual boundary points Q_1-Q_5 of sample points P_1-P_5

In detail, as shown in Fig.2, the following steps are involved in the proposed method.

(1) Discretization of the CAD model: The CAD model is discretized into a triangular mesh model with a set of sample points under certain accuracy.

(2) Space division: The bounding box of the model is divided into grid cells of equal sizes for the following divide-and-conquer based computation of MA points. Here the edge length of each grid cell is called *gl* for short.

(3) Estimation of normal directions of boundary points: The normal directions of points on all boundary triangles are estimated by an interpolation-based method.

(4) Computation of MA points: Divide all sample points into several point sets according to their medial radii and their MA points are computed in an iterative manner. In each tracing round *i*, the MA points of sample points in the current range [i×gl, i×gl+gl) are computed by Steps (5)-(6) until the MA points of all sample points have been obtained.

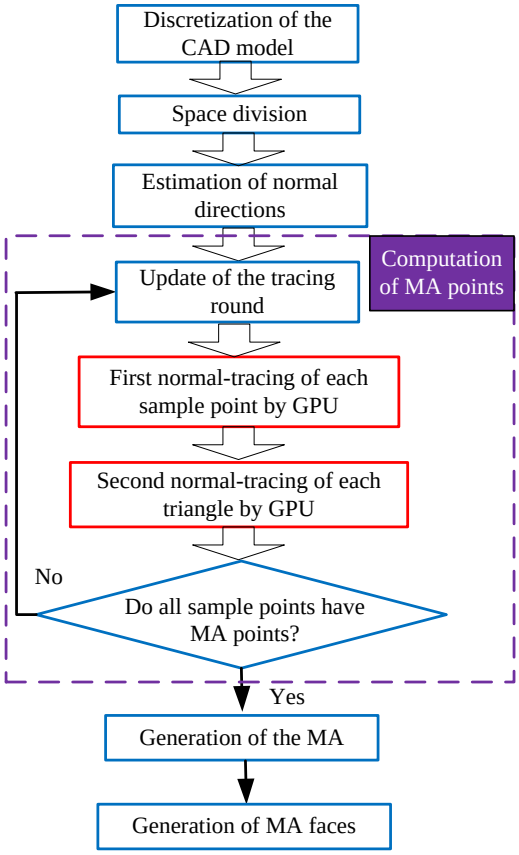


Fig. 2 Overview of the poposed method

(5) First normal-tracing of each sample point by GPU: For each sample point for which the MA point computation has not been completed, trace its normal segment in the current range. A GPU thread is allocated for each sample point during this process.

(6) Second normal-tracing of each triangle by GPU: For each triangle, trace the segment of its normal ray cluster in the current range. The MA points are computed by searching the possible pairs of sample point and triangle using the divided grid cells. A GPU thread is allocated for each triangle during this process.

(7) Generations of the MA and MA faces: Transform the MA points into the resultant MA by using the topological connectivities of their corresponding sample

points. Then the MA faces are extracted from the resultant MA.

4 Pretreatment

As the computational complexity of generating an accuate MA for a CAD model is high, the proposed method focuses on obtaining an approximated one. The pretreatment contains the discretization of the CAD model, space division and the estimation of normal directions.

4.1 Discretization of the CAD model and space division

The model is first descretized into a triangular mesh model by Delaunary triangulation [46]. Here, the number of sample points is determined by the user to control the accuracy and the complexity of the proposed method.

For each sample point, its MA point is found by searching the corresponding triangle. To reduce the searching range, the divide-and-conquer algorithm is used. First, the range of the model is divided into some grid cells. And then, the searching range of the corresponding triangle for each sample point is limited in corresponding grid cells only.

In detail, the space of the bounding box of the model needs to be divided into several grid cells by slicing it on the *x*, *y* and *z* coordinate axes. Each grid cell is a cub with edge size *gl*. As shown in Fig.3, 2D model *M* is divided into 36 grid cells.

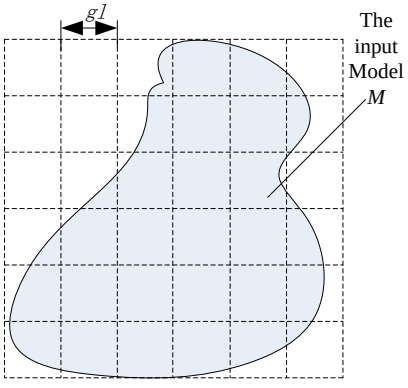


Fig. 3 Space division

4.2 Estimation of normal directions of boundary points

To compute the MA point for each sample point according to the definition of MA, normal directions of boundary points needs to be obtained first. However, the number of boundary points is infinite and it is difficult to obtain the accurate values for all of them especially when some boundary surfaces are complicated. It is noteworthy that normal directions may change with a discontinuous feature on points on a triangle edge after model meshing. Therefore, normal directions of boundary points should be reestimated to achieve their continuities.

To eliminate the discontinuity of normal directions of points within a triangle, an interpolation-based method is proposed in this study. As shown in Fig.4, point *P* locates on a triangle whose vertices are points *A*, *B* and *C* with normal directions N_A, N_B and N_C, respectively. Suppose the areas of triangles follow the equation $Area_{PBC}$: $Area_{PAC}$: $Area_{PAB}$ = *x*: *y*: *z* and *x* + *y* + *z* = 1. Then, by using the barycentric coordinates, the coordinate of point *P* can be represented by points *A*, *B*, *C* as follows: *xA*+*yB*+*zC*. Using an interpolation equation, the normal N_P of point *P* is estimated as follows:

$$N_P = xN_A + yN_B + zN_C \tag{1}$$

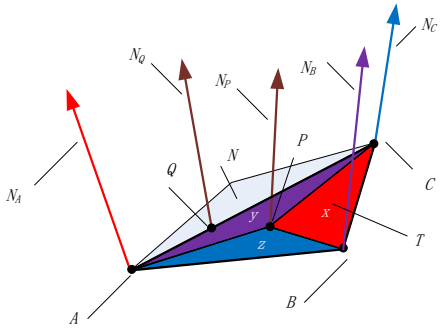


Fig. 4 The estimation of the normal direction of point *P* in triangle *T*

It is not difficult to see that according to Eq. (1), N_P is equal to N_A, N_B or N_C if point *P* locates on *A*, *B* or *C*, respectively. Eq. (1) can also keep the continuities of the normal directions of points within a triangle. Suppose point *P*' is a boundary point near point *P*. The coordinate of point *P*' can be represented by $(x+\varepsilon_x)A+ (y+\varepsilon_y)B+ (z+\varepsilon_z)C$ where ε_x, ε_y, ε_z are close to 0. Using Eq. (1), the normal direction of point *P*' is obtained as $(x+\varepsilon_x)N_A+ (y+\varepsilon_y)N_B + (z+\varepsilon_z)N_C$, which is near the direction of its neighbor point *P*. Therefore, the

normal directions of points within a triangle are continuous.

5. Computation of MA Points

Using the normal directions of boundary points, a novel parallel dual-normal-tracing (DNT) algorithm is proposed in this work to serve the purpose of iteratively obtaining an MA point for each sample point based on the divide-and-conquer algorithm. The space division has been conduced to limit the searching range of the corresponding triangle for each sample point in corresponding grid cells only, as stated in section 4.

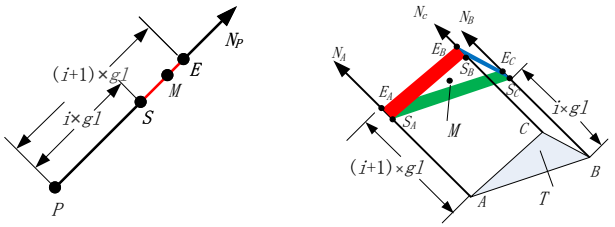
5.1 Introduction of the DNT algorithm

The DNT algorithm is an iterative process that involves several rounds of tracing (the first one marked as the 0^{th} round). In the *i*th round, the sample points whose medial radii are in the range [$i\times gl$, (i+1)×gl) will generate their MA points by two normal tracings. This range is called the ***current range*** of round *i*. After each round, the number of the involved sample points will get smaller since the MA points of some sample points have already been computed in this round and they do not need to be involved in the next round. After a few rounds of executing the procedure, all sample points get their MA points and the algorithm ends. For each round, these two normal tracings are called the first normal tracing and the second normal tracing, respectively.

5.1.1 First normal tracing of each sample point by GPU

The first normal tracing of a sample point is to compute its MA point by tracing its normal ray iteratively. For each sample point *P*, its MA point *M* must locate on its normal ray N_P. In the *i*th round, since the medial radius of sample point *P* is required to be in the current range, the MA point *M* must locate on the red normal segment *SE* which starts at point *S* where *PS* is $i\times gl$ and ends at point *E* where *PE* is (i+1)×gl , as shown in Fig.5a.

To reduce the searching area of MA point *M*, grid cells are used to trace the normal segment of sample point *P*. As normal segment *SE* contains MA point *M*, only the grid cells that intersect with normal segment *SE* in the current range may contain MA point *M*. Here, since the length of normal segment *SE* is *gl*, its starting point *S* and ending point *E* must locate in the same grid cell or two face/edge/point-neighbor grid cells. For the latter case, as shown in Fig.6a-c, the grid cells that contain the starting point *S* and the ending point *E* are marked in purple and red respectively. At most 2, 3, and 4 grid cells will intersect with the normal segment for the face, edge, and point neighbor cases, respectively. Those grid cells record sample point *P* as their ***candidate sample point*** in a linked list. It means that only the grid cells that have candidate sample point *P* may contain the MA point *M* of sample point *P*. A linked list is used to store candidate sample points for a grid cell for the purpose of minimizing the memory cost of a grid cell.

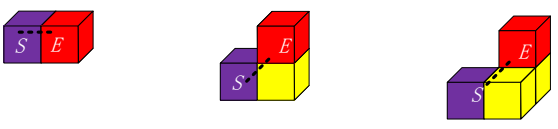


(a) Normal tracing of sample point *P*

(b) Normal tracing of triangle *T*

Fig. 5 Normal tracings of a sample point and a triangle respectively

The first normal tracing of each sample point has no mutual dependency and can be implemented in parallel. To speed up the normal tracings of sample points, a GPU thread is allocated to the normal tracing of each sample point in this study. Considering that sample points in different GPU threads may store them as candidate sample points in the same grid cell in the same time, the operation of storing store a candidate sample point in a grid cell is atomic.



(a) Face-neighbor (b) Edge-neighbor (c) Point-neighbor

Fig. 6 Three types of neighbor grid cells

5.1.2 Second normal tracing of each triangle by GPU

After the first normal tracing of sample point *P* in the *i*th round, its MA point *M* is limited in some grid cells which record it as its candidate sample point. However, to compute

MA point M, it is necessary to search the dual triangle T of point P. Since every triangle can possibly be the dual triangle T of sample point P, all the triangles need to be checked.

For any point on a triangle T, the MA point M must locate in the normal ray cluster of triangle T. During round i, as the medial radius of any point on triangle T is required to be in the current range, MA point M of the point must locate in the normal segment cluster which starts at $S_AS_BS_C$ where $AS_A/BS_B/CS_C$ is $i\times gl$, and ends at $E_AE_BE_C$ where $AE_A/BE_B/CE_C$ is $(i+1)\times gl$, as shown in Fig.5b. In Fig.5b, the normal segment cluster is a triangular prism bounded by three quadrangles $S_AS_BE_AE_B$, $S_AS_CE_AE_C$ and $S_BS_CE_BE_C$ in red, green and blue respectively and two triangles $S_AS_BS_C$ and $E_AE_BE_C$. Only the grid cells that intersect with this normal segment cluster may contain MA point M. To determine the intersected grid cells, the bounding box of the normal segment cluster is computed and then the grid cells that intersect with the bounding box can be obtained as them.

To compute the bounding box of a normal segment cluster, the representation of the MA point M in the normal segment cluster is given according to Eq. (1) as follows.

$$M = P + N_P r = xA + yB + (1-x-y)C + (xN_A + yN_B + (1-x-y)N_C)r \quad (2)$$

In Eq. (2), point P is the corresponding boundary point of medial point M on triangle T. Obviously, x, y and 1-x-y are all non-negative and r is in the range $[i\times gl, (i+1)\times gl)$. By using these restrictions in three coordinates, the range of medial point M in three coordinates can be determined and thus the bounding box of the red normal segment cluster can be obtained.

During this process, triangle T is called a ***candidate dual triangle*** of these grid cells**.** It means that only these grid cells may contain the MA point whose dual triangle is triangle T. And then, by checking pairs of candidate sample point and candidate dual triangle of a grid cell in these grid cells, MA points in this grid cell can be generated as described in the following subsection.

Since there is no coupling relationship between the second normal tracings of different triangles, a GPU thread is allocated to the normal tracing of each triangle.

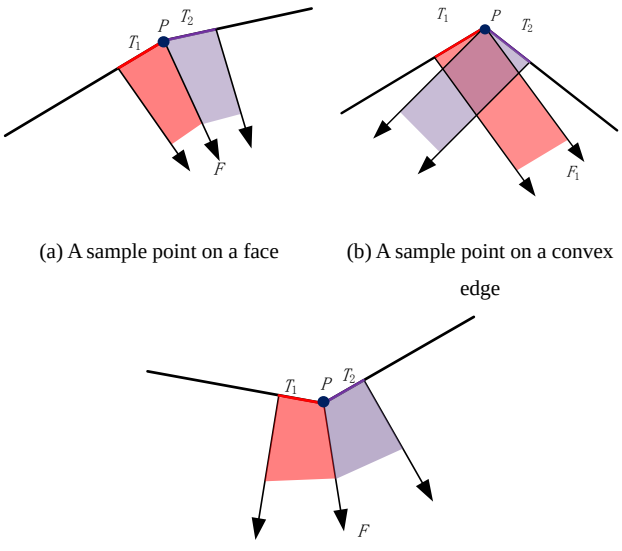
5.1.3 Additional definitions for the second normal tracing of each triangle

To perform the second normal tracing of each triangle, the normal directions of vertices of each triangle should be uniquely decided. However, a sample point on a boundary edge may have multiple normal directions.

The normal directions of most boundary points, for example, the normal direction of any boundary point P on a boundary face, can be determined using its original one before model meshing. Take sectional views in Fig.7a-c as examples. As shown in Fig.7a, point P locates on a boundary face of the original model, and its unique normal direction can be used by the normal clusters of triangles T_1 and T_2. Otherwise, point P must locate on a boundary edge of the original model. The value of its normal direction is not unique and can be computed by the following definitions.

Definition 1: The normal direction of a sample point on a convex edge equals to that of the triangle that contains this sample point when computing the normal cluster of this triangle.

Definition 2: The normal direction of a sample point on a concave edge equals to the average normal direction of the triangles that share this sample point.



(a) A sample point on a face (b) A sample point on a convex edge

(c) A point on a concave edge

Fig. 7 Different conditions of normal directions of sample point P

As shown in Fig.7b, triangles T_1 and T_2 share a convex edge which contains sample point P. When computing the

normal cluster of T_1 or T_2, the normal direction of sample point P is that of triangle T_1 or T_2. As shown in Fig.7c, triangles T_1 and T_2 share a concave edge which contains sample point P. The normal direction of sample point P is the average normal direction of triangles T_1 and T_2 that share point P.

5.2 Generation of MA points based on DNT

After the second normal tracing of triangle T in round i, an intersected grid cell will have some pairs of candidate sample point and candidate dual triangle, as stated above. For each intersected grid cell G, it has recorded triangle T as its candidate dual triangle. Meanwhile, it may also record some candidate sample points in the first normal tracings of these sample points. For each point P of these sample points, since MA point M is the center of the medial ball tangent to point P and the dual triangle T, the following equation holds:

$$M = P + N_P \times r = Q + N_Q \times r \tag{3}$$

In this equation, r refers to the medial radius of MA point M while point Q refers to the dual boundary point on dual triangle T. By replacing N_Q with Eq. (1), Eq. (3) can be rearranged as Eq. (4).

$$r = \frac{P - Q}{N_Q - N_P} = \frac{P - (xA + yB + (1 - x - y)C)}{xN_A + yN_B + (1 - x - y)N_C - N_P} \tag{4}$$

As Eq. (4) holds in all three coordinates, it can be converted to three equations with three unknowns r, x, y as the following forms.

$$r = (a_1 \times x + b_1 \times y + c_1) / (d_1 \times x + e_1 \times y + f_1) \tag{5}$$

$$r = (a_2 \times x + b_2 \times y + c_2) / (d_2 \times x + e_2 \times y + f_2) \tag{6}$$

$$r = (a_3 \times x + b_3 \times y + c_3) / (d_3 \times x + e_3 \times y + f_3) \tag{7}$$

In the equations above, a_i, b_i, c_i (i = 1, 2, 3) are all constant values obtained by Eq. (4) in three coordinates, respectively. The value of x can be calculated in Eq. (8) and Eq. (9) by Eq. (5)&(6) and Eq. (5)&(7), respectively.

$$x = \frac{(f_1 \times r - c)(b_2 - e_2 \times r) - (f_2 \times r - c_2)(b_1 - e_1 \times r)}{(a_1 - d_1 \times r)(b_2 - e_2 \times r) - (a_2 - d_2 \times r)(b_1 - e_1 \times r)} \tag{8}$$

$$x = \frac{(f_1 \times r - c)(b_3 - e_3 \times r) - (f_3 \times r - c_3)(b_1 - e_1 \times r)}{(a_1 - d_1 \times r)(b_3 - e_3 \times r) - (a_3 - d_3 \times r)(b_1 - e_1 \times r)} \tag{9}$$

Eq. (8)-(9) can be further converted to Eq. (10) as a quartic equation with only one unknown r.

$$\begin{aligned} &((a_1 - d_1 \times r)(b_2 - e_2 \times r) - (a_2 - d_2 \times r)(b_1 - e_1 \times r)) \times ((f_1 \times r - c) \\ &(b_3 - e_3 \times r) - (f_3 \times r - c_3)(b_1 - e_1 \times r)) = ((a_1 - d_1 \times r)(b_3 - e_3 \times r) \\ &-(a_3 - d_3 \times r)(b_1 - e_1 \times r)) \times ((f_1 \times r - c)(b_2 - e_2 \times r) - \\ &(f_2 \times r - c_2)(b_1 - e_1 \times r)) \end{aligned} \tag{10}$$

Any quartic equation with one unknown has no larger than 4 solutions and each solution can be represented by a general constant formula accurately and conveniently [47]. It is noteworthy that the number of solutions of r may be larger than one, but only the medial radii which are in the current range are chosen.

After assigning each r of the possible medial radii above into Eq. (5)-(6), x, y are both solved. Using the interpolation equation, if x, y and z are all non-negative, the dual boundary point Q of sample point P will locate on dual triangle T and MA point M of sample point P is computed. Although a sample point P may generate more than one ball by adopting different r obtained in the quartic equation, only the ball with the smallest radius is adopted. As shown in Fig.8, for sample point P, its tangent balls B_1 and B_2 are both tangent to the boundary on Q_1 and Q_2 respectively. However, only ball B_1 with the smallest radius is considered as the medial ball of point P since it is the smallest ball that is tangent to two boundary points according to the property of medial ball.

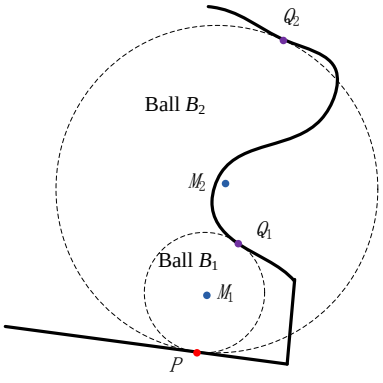


Fig. 8 Multiple tangent balls for simple point P

Based on the above analysis, the proposed DNT algorithm is described as a pseudo-code scheme shown in Fig.9.

```
//Input: triangles and sample points, length of grid edge gl
//Output: MA points of all sample points
set the round count i to 0;
while at least one sample point without MA point exists
  set the current range as [i×gl, (i+1)×gl);
  for each sample point P /*The first normal-tracing*/
    obtain the grids which intersect with the normal
    segment of boundary sample point P in the current
    range;
    for each intersected grid V
      record sample point P as a candidate sample point of
      grid V;
    end for
  end for
  for each triangle T /* The second normal-tracing */
    compute the grids that intersect with the normal
    segment cluster of triangle T in the current range;
    for each intersected grid V
      for each candidate sample point P of grid V
        compute the medial radius r of sample point P
        and triangle T;
        for each solved r
          if r is in the current range and is smaller than
          the previous medial radius of sample point P
            record r as the new medial radius of sample
            point P;
          end if
        end for
      end for
    end for
  end for
  i is updated to i+1
end while
```

Fig. 9 The pseudo code of the DNT algorithm

5.3 Theoretical analysis of the DNT algorithm

In this section, the correctness of the DNT algorithm is analyzed through a mathematical deduction. For any sample point P whose medial radius is r, suppose r is in the range $[i\times gl, (i+1)\times gl)$. It only needs to prove that in the ith round, MA point M of sample point P can be computed.

Since r is in the current range of the ith round, MA point M locates on the normal segment of sample point P which starts at $i\times gl$ and ends at $(i+1)\times gl$, as shown in Fig.5*a*. After the grid cells that intersect with this normal segment are identified by the first normal tracing of sample point P, it is confirmed that MA point M is in these grid cells. Here, sample point P is recorded as a candidate sample point by these grid cells.

As the dual boundary point Q of sample point P can locate on any triangle (e.g. T), all the triangles need to be checked for searching dual boundary point Q. Since r is in the current range of the ith round, it is confirmed that MA point M also locates in the normal segment cluster of triangle T which starts at $i\times gl$ and ends at $(i+1)\times gl$, as shown in Fig.5*b*. After the grid cells that intersect with this normal segment cluster are searched by the second normal tracing of triangle T, it is confirmed that MA point M must locate in these grid cells. Then, triangle T is recorded as a candidate dual triangle by these grid cells. Thus, the grid cell which contains MA point M must have both candidate sample point P and candidate dual triangle T. In this grid cell, MA point M can be generated using Eq. (10) with the information of sample point P and candidate dual triangle T. Therefore, any sample point P with arbitrary medial radius can generate its MA point M using the DNT algorithm in a certain round.

6. Generations of the MA and MA Faces

In this section, the method of generating the MA using MA points is introduced first. And then, the procedure in which MA faces are identified from the MA is detailed.

6.1 Generation of the MA

In a mesh model, for the three vertices of each triangle T, there are three corresponding MA points on the resultant MA. An MA triangle can be obtained by connecting these three MA points, which can then be considered as the corresponding MA triangle of triangle T. As shown in Fig.10, sample points P_1-P_5 are vertices of three green triangles and their MA points are M_1-M_5, respectively. By connecting MA points T_1-T_5, using the topological connectivities of their corresponding sample points P_1-P_5, three green MA triangles are generated.

The generated MA triangles have a one-to-one mapping relationship with the boundary triangles. Since the boundary triangles are connected, the generated MA triangles are also connected and the original topological connectivities of boundary triangles is preserved in the MA triangles.

6.2 Generation of MA faces

The MA of a CAD model is formed by some CAD faces called ***MA faces***. The MA points on an MA face are all

generated by two boundary points from one or two boundary faces. Therefore, to identify those MA faces from the MA, all MA points should first be marked by the one or two boundary faces of their corresponding sample points and corresponding triangles. As shown in Fig.11, the MA points of a square can be grouped into four MA faces. For MA points of these four faces, their corresponding sample points and dual triangles are on Faces 1&2, Faces 1&4, Faces 2&3 and Faces 3&4 respectively. Those MA faces are named by the names of their corresponding boundary faces, such as MA_{12}, MA_{14}, MA_{23} and MA_{34}.

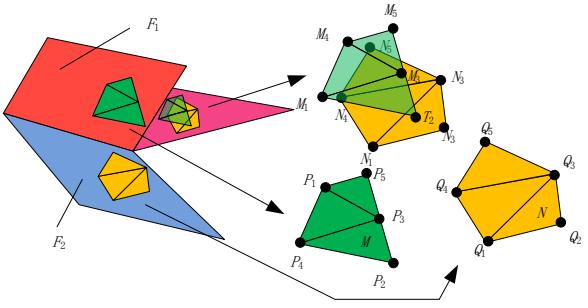


Fig. 10 MA generation

Generally, an MA point belongs to MA face MA_{ij} where i and j refers to the face numbers of the two corresponding boundary points of the MA point. After identifying the MA faces of an MA point by the rule above, the MA face of an MA triangle is determined by the MA faces of its three MA vertices which are usually on the same MA face. As shown in Fig.11, the two MA vertices of MA triangle T_1 are both on MA_{12}, therefore MA triangle T_1 belongs to MA_{12}. However, for an MA triangle on the common boundary of different MA faces, it may have MA vertices of two or three MA faces. Specifically, if an MA triangle has MA vertices of different MA faces, all those MA faces are recorded as its MA faces. As shown in Fig.11, the MA faces of MA triangle T_2 are both MA_{12} and MA_{14}.

The generation of MA faces is applicative for a CAD model with parametric surfaces and freeform surfaces. However, it is not effective for those triangular mesh models without original boundaries, though the resultant MA can be generated for them.

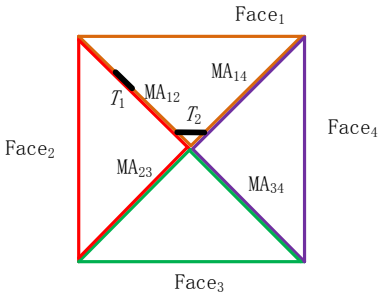


Fig. 11 Generation of MA faces

7. Analyses of the Complexity, Robustness, Quality and Applicability

To illustrate the performance of the proposed method, the time complexity and space complexity of the DNT algorithm is analyzed first. Using the time complexity, a strategy to optimize the time cost of DNT method is proposed. In addition, some methods are given to improve the robustness of the proposed method. Finally, the applicability of the proposed method is discussed.

7.1 Complexity analysis

7.1.1 Time complexity of the DNT algorithm

The major time cost of the proposed method is associated with the iterative DNT algorithm. In the 0^{th} round, the time cost is represented as follows.

$$T_1 = Nt_1 + (2N-4)(t_2 + kt_3) \approx Nt_1 + 2N(t_2 + kt_3) \quad (11)$$

In Eq. (11), N refers to the number of sample points and t_1 refers to the time cost of the first normal tracing of a sample point. According to Euler's formula, assume there are f triangles and it can be obtained that N-1.5f+f=2, as each triangle have three half edges. Therefore, there are 2N-4 triangles. Additionally, t_2 refers to the time cost of the second normal tracing of a triangle; k refers to the average number of iteration rounds of computing MA points for a triangle; and t_3 means the time cost of computing an MA point from a pair of candidate sample point and candidate triangle.

To further analyze the time complexity, the 0^{th} round of

the algorithm is analyzed first. Suppose the resolution gridization is $r\_x \times r\_y \times r\_z$ on three coordinate axes and then there are about a total of abr^3 grid cells. Here $r = r\_x$, and $r\_y = ar$, $r\_z = br$, assuming that r_x is the smallest one in r_x, r_y and r_z. In the first normal tracing, the normal segment of each sample point intersects with two grid cells. Therefore the average number of candidate sample points of a grid cell is $2N/abr^3$. In the second normal tracing, the normal segment cluster of each triangle intersects with 2 grid cells in average and each grid cell has $k=2N/abr^3$ candidate sample points as analyzed above. For each triangle, the MA points are generated from the intersected grid cells by checking their candidate sample points. Finally, the complexity formula in Eq. (11) is rewritten as follows.

$$T_1 = Nt_1 + 2N(t_2 + 4Nt_3 / abr^3) \tag{12}$$

Since the gridization resolution on the shortest coordinate axis is r, the algorithm will end in the [r/2-1]-th round. In each round of iteration, the sample points that have their MA points obtained will not be involved. The average time for computing the MA points of $2N/r$ sample points is roughly one round of iteration. Therefore, in the ith round, the number of sample points involved in the first normal tracings is $(r\text{-}2i)N/r$ while the number of the involved triangles in the second normal tracings is always $2N$. Finally, the total time cost can be deduced by using Eq. (12) as follows.

$$\begin{aligned} T &= \sum_{i=0}^{r/2-1} ((r-2i)Nt_1 / r + 2N(t_2 + 4(r-2i)Nt_3 / abr^4)) / p \\ &\approx \sum_{i=0}^{r/2-1} (2iNt_1 / r + 2N(t_2 + 8iNt_3 / abr^4)) / p \\ &\approx (Nrt_1 / 4 + 2N(rt_2 / 2 + Nt_3 / abr^2)) / p \end{aligned} \tag{13}$$

Since the first normal tracing of each sample point and the second normal tracing of each triangle are both implemented by a GPU in parallel, the time complexity in Eq. (13) is divided by p which refers to the number of GPU threads for parallel processing.

7.1.2 Adaptive division

The representation in Eq. (13) indicates that the time complexity is not constant because it has an unknown r which may impact the final time cost of the DNT algorithm. To minimize the time cost, r should be adjusted to a proper value in the division of the mesh model which is called a procedure of ***adaptive division***. As N is a constant for a given mesh model while r is the only variable, the derivative of the time cost with respect to r in Eq. (13) can be represented as

$$dT / dr = (Nt_1 / 4 + Nt_2 - 4N^2 t_3 / abr^3) / p \tag{14}$$

The minimum value of the time cost can be obtained by solving r when the derivative of the time cost is zero as follows.

$$r = \sqrt[3]{(16t_3 N / ab(t_1 + 4t_2))} \approx 5\sqrt[3]{N / ab} \tag{15}$$

In Eq. (15), t_1, t_2 and t_3 all have constant values that can be estimated in the experiments and $5\sqrt[3]{N/ab}$ is obtained as the value of r. Specifically, t_3 is much larger than t_1 and t_2 because solving a quartic equation is more expensive in terms of time cost than the two normal tracings. It can be deduced that when the gridization resolution on the shortest coordinate axis for the proposed method is $5\sqrt[3]{N/ab}$, the adaptive division is conducted and the best performance of the proposed method can be obtained.

As r is $5\sqrt[3]{N/ab}$, by using Eq. (13), it can be concluded that

$$T = N^{4/3}(t_1 / 4 + 2(t_2 / 2 + t_3 / ab)) / p \tag{16}$$

As t_1, t_2, t_3, a, b and p are all constants, the time complexity of the proposed method is O ($N^{4/3}$).

7.2 Robustness analysis

The boundary of the original CAD model may contain some noise faces which are usually of small area. Therefore, before model meshing, small faces needs to be merged into a nearby normal face by some rules stated in the authors' previous work [6]. Initially, each boundary face is a face unit. Then, the small face units are merged with their neighbor face units until every face unit contains adequate area.

7.3 MA Quality

To evaluate the resultant MA, its quality needs to be analyzed. As the MA faces of the resultant MA are generated using MA points, the quality of MA points needs to be evaluated first. In detail, the errors of MA points in the

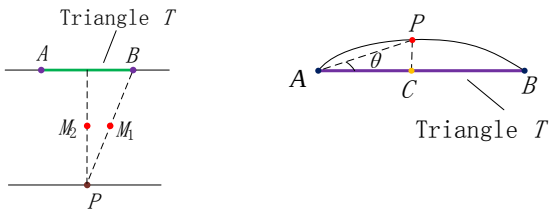
proposed method and two other methods are listed for comparisons. Furthermore, the trade-offs between the quality and the efficiency of the proposed method is given.

7.3.1 Analysis and comparisons of MA quality

The MA quality in the proposed method is analyzed in two conditions: for plane boundaries and for curved boundaries. Here, boundary error of a boundary point refers to the distance between a boundary point on the triangle and its original position while MA error of a sample point refers to the distance between its true MA point and its resultant MA point.

In the first condition, the normal direction in the first normal tracing and the normal cluster in the second normal tracing are all equal to the ones of the original surfaces. Meanwhile, the boundary error of any boundary point on the planar surfaces is 0. Therefore, according to the definition of MA, the MA error of any boundary point on the planar surfaces is 0. For example, the resultant MA point M_2 of sample point P in Fig.12a is exactly the true MA point by solving the equations of point P and triangle T.

In the second condition, first the boundary error of a curved boundary point is evaluated. As shown in Fig.12b, points A is a vertex on triangle T while point P is an arbitrary point on the original curved boundary of triangle T. Edge AB is obtained by cutting triangle T using its vertical plane that has points A and P. The boundary error of point P can be calculated as follows.



(a) MA Error of sample point P on a planar surface

(b) Surface error of point C on Triangle T

Fig. 12 Evaluation of MA error

$$PC = AC\tan\theta < \varepsilon\tan\theta < \varepsilon \tag{17}$$

Here, ε refers to the average distance between sample points while θ refers to $\angle PAC$, which is usually smaller than $\pi/4$. Since MA points are generated using boundary triangles according to the definition of MA, the MA error of the proposed method is O (ε), as shown in Table 1.

As stated in the distance dilation method [48] and the medial axis points method [10], their calculation errors are O (VS) and O (ε), as shown in Table 1. Here, VS refers to the size of voxels. Therefore, the quality of our method is better than that of current methods when computing the MA of a CAD model which usually contains many plane surfaces. For example, in the medial axis points method [10], MA point of sample point P on a planar surface is generated by searching its corresponding sample point B, leading to an error between the resultant MA M_1 and the true MA M_2.

Table 1. Qualities of MA points of different methods

Method	MA error
Ours	0 for plane surfaces, O (ε) for curved surfaces
The distance dilation method	O (VS)
The medial axis points method	O (ε)

7.3.2 Trade-off between the quality and the efficiency

To study the trade-off between the quality and the efficiency, the relationship between the error of the resultant MA and the time cost of the proposed method needs to be discussed.

As analyzed in section 7.1.2, time cost T of the proposed method is $k_1N^{4/3}$ while the MA error E of the resultant MA is $k_2\varepsilon$ for curved surfaces, as given in section 7.3.1. Here, N refers to the number of sample points while ε refers to the average distance of sample points, k_1 and k_2 are both constant. In the close 3D space with N sample points whose average distance is ε, it can be concluded that the value of its volume V is $N\varepsilon^3/8$. Therefore, the relationship between T and E is

$$(T/k_1)^{3/4}E^3 = 8Vk_2^3 \tag{18}$$

Then, the trade-offs between the quality and the efficiency can be deduced as the following equation, where $Cons$ refers to a constant.

$$T \times E^4 = Cons \tag{19}$$

7.4 Applicability

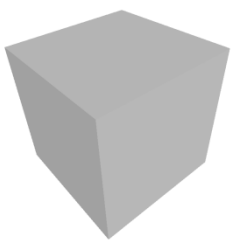
In the first step of the proposed method, the input CAD is descritized into a triangular mesh model. Therefore, this method can also be applied to a triangular mesh model. Also, a model of another type can be transferred into a triangular mesh model and this method can be conducted to generate its MA. In the following experiments, the proposed method will be applied to both CAD models and triangular mesh models.

8. Implementation

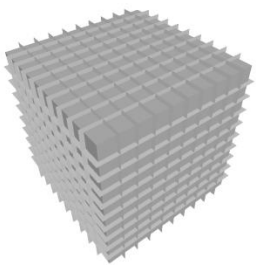
The proposed method has been implemented with visual studio 2010 and CUDA 6.5. The hardware is a notebook with Intel i7 4700 CPU and NVIDIA GTX765M GPU. To demonstrate the efficacy and efficiency of the proposed method, four groups of experiments are conducted as follows. The first group is used to explain the detailed steps of the proposed method. The second group applies the proposed method in different gridization resolutions to test the differences in the execution times. In the third group, detailed comparisons between different methods are described to systemtically compare the performances of these methods. Finally, some examples with different numbers of triangles are given to confirm the correctness of the proposed method with further evidence. To maximize the performance, the adaptive division is used for the third and forth groups.

8.1 Demonstration of the whole process

To help the readers understand the whole process of the proposed method, a cube is used in the first experiment. It is descritized into a triangular mesh with 818 sample points, as shown in Fig.13a. The gridization resolution is set to 12×12×12 as shown in Fig.13b. To compute the MA points, the DNT algorithm is applied to the cube. As shown in Fig.13c, in the 0^{th} round, MA points of a few sample points are computed. In the fifth round, MA points of all sample points are computed and the DNT algorithm ends with a result shown in Fig.13d. Then, the resultant MA is generated as shown in Fig.13e. Finally, the MA faces are generated as shown in Fig.13f with different colors. This convention of the resultant MA is used all through Section 8. Since a NVIDIA GTX765M GPU with 768 streaming processors is used, the parallelism of the experiments in this paper is 768.



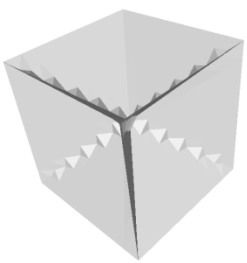
(a) Model 1



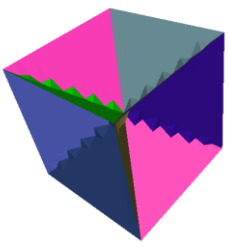
(b) Model division

(c) The 0^{th} round of the DNT algorithm

(d) The fifth round of the DNT algorithm



(e) The MA



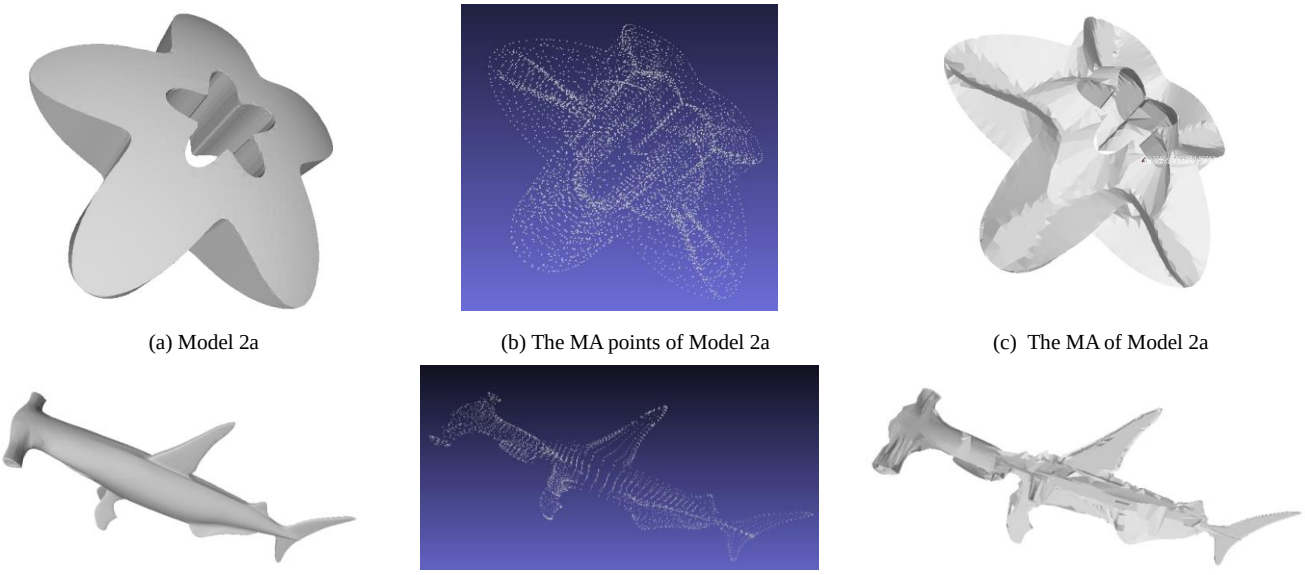
(f) The MA faces

Fig. 13 Illustration of the MA generation procedure for Model 1

8.2 Illustration of the effect of resolution

The execution time of the proposed method depends on the gridization resolution as analyzed in subsection 7.1. Here the gridization resolution refers to that on the shortest coordinate axis. To compare the differences of execution times under different resolutions, another two models, named Model 2a and 2b in Fig.14a and 14d are used in this experiment. As shown in Fig.14b and 14e, the resultant MA points of Models 2a and 2b are computed by the DNT algorithm. Finally, the resultant MAs of Models 2a and 2b are generated as shown in Fig.14c and 14f.

Although the resultant MA of the proposed method for a single model is the same, the execution time varies by using different resolutions. As shown in Table 2, for Model 2a, when the resolution is 50, the execution time is the smallest. In Model 2a, a and b are both 2 while N is 5188. According to Eq. (15), the resolution of the adaptive division is 53, which is similar to this value. For Model 2b, a similar conclusion can be deduced as shown in Table 2.



(a) Model 2a (b) The MA points of Model 2a (c) The MA of Model 2a

(d) Model 2b (e) The MA points of Model 2b (f) The MA of Model 2b

Fig. 14 The MA points and MA of Models 2a and 2b

Table 2. Execution times of Models 2a and 2b

Model No.	Sample point No.	Resolution	Time(ms)	Model No.	Sample point No.	Resolution	Time(ms)
2a	5188	10	334	2b	4820	10	333
2a		20	181	2b		20	177
2a		30	134	2b		30	130
2a		40	117	2b		40	114
2a		50	114	2b		50	109
2a		60	115	2b		60	126
2a		70	120	2b		70	127
2a		80	151	2b		80	141

8.3 Comparisons of different methods

To demonstrate the performance of the proposed method, the following four representative algorithms are implemented and compared.

(1) The discrete scale axis [42] which can generate a stable MA through mesh operation;

(2) The distance dilation method [48] which can generate the MA through operations of voxels;

(3) The parallelized double-queue-distance-dilation based method [7] which can generate the MA by using multi-CPU.

(4) The parallelized medial axis points [10] which can generate the MA by using GPU.

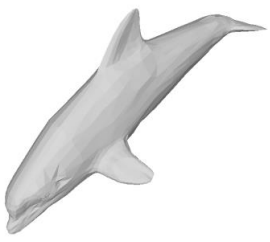
In this section, the proposed method is implemented by using both GPU and CPU to make various comparisions with these methods.

8.3.1 Comparison with the discrete scale axis

A representative work of MA generation was proposed by Miklos *et al* [42]. In this method, a stable medial axis, i.e. the discrete scale axis, was proposed. However, the major cost of the discrete scale axis is in its pretreatment in which the input model is converted to balls.

Model 3a in Fig.15a is used to compare the results between this method and the method proposed in this work with GPU/CPU. The parameters of the discrete scale axis are δ=0.01, s=1.1 and a comparison of outputs is shown in Fig.15b-c as well as in Table 3. It can be seen that the method proposed in this work is more efficient. Comparisons in different parameters show similar conclusions.

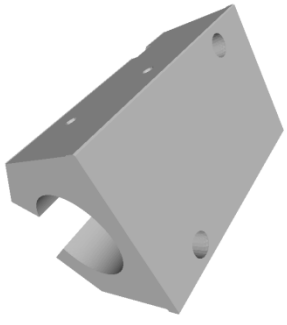
Although the proposed method with a GPU has a much larger degree of parallization than that with a CPU, a CPU thread is actually much faster than a GPU thread. Therefore, efficiency improvement achieved by using GPU is generally in the scale of 5-10 times, as shown in Table 3.



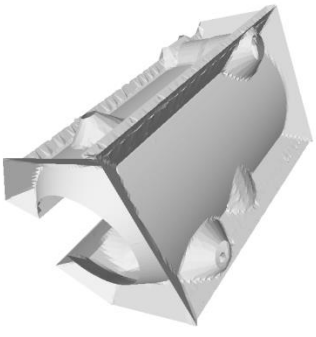
(a) Model 3a

(b) The MA generated using the proposed method for Model 3a

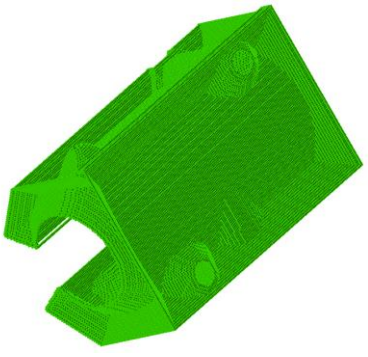
(c) The discrete scale axis for Model 3a



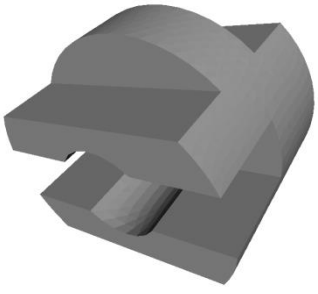
(d) Model 3b



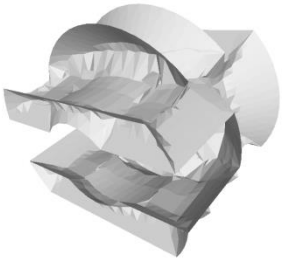
(e) The MA generated using the proposed method for Model 3b



(f) The MA of the dilation method and the DQDD-based method for Model 3b



(g)Model 3c



(h) The MA generated using the proposed method for Model 3c

(i) The MA points for Model 3c

Fig. 15 Comparisons of outputs from different methods

8.3.2 Comparison with the distance dilation method and the parallized DQDD-based method

The distance dilation method [48] and the parallized DQDD-based method [7] both generate MAs represented by voxels. However, the latter one uses multi-CPU to further divide the computation load of MA generation. The major cost of these two methods is in their common pretreatment methods, i.e. voxelization.

Model 3b in Fig.15d is used to compare the results obtained using these two methods and those obtained using the proposed method. The comparison is shown in Fig.15e-f as well as in Table 3. It can be seen that again the proposed method is more efficient.

Table 3. Execution times of Models 3a-c

Model No.	Method	Main processor	Number of sample points/ boundary voxels	Time of pretreatment(ms)	Time of MA generation (ms)	Total time(ms)	Degree of parallelism
3a	Ours	GPU	2069	0	93	93	768
3a	Ours	CPU	2069	0	618	618	1
3a	The discrete scale medial axis	CPU	2069	14831	2659	17490	1
3b	Ours	GPU	134475	0	2861	2871	768
3b	Ours	CPU	134475	0	16381	16429	1
3b	The distance dilation method	CPU	138852	520239	1467	521706	1
3b	The DQDD-based method	Multi-CPU	138852	520239	468	520707	4
3c	Ours	GPU	2663	0	84	85	768
3c	Ours	CPU	2663	0	577	579	1
3c	The medial axis points	GPU	2663	0	79	79	768
3c	The medial axis points	CPU	2663	0	612	612	1

8.3.3 Comparison with the parallelized medial axis points

Another representative work of MA generation based on GPU generates medial axis points using nearest neighbors and the normal field [10]. As the result of this method is a set of points without extracted MA faces, its applications tend to be limited. What's more, for each sample point, the corresponding sample point is searched in their method, while in our method, the corresponding triangle is searched. The MA error of our method can be elimated for planer surfaces.

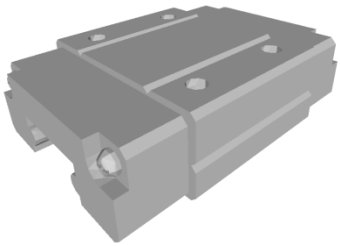
Model 3c in Fig.15g is used to compare the results obtained using this method and those obtained using the proposed method. A detailed comparison is shown in Fig.15h-i as well as in Table 3. It can be seen that the time costs of these two methods are similar but the results achieved by the proposed method contain MA faces.

8.4 Illustration of the effects of triangle number

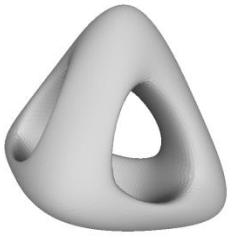
To further demonstrate the performance of the proposed method, Models 4a-c shown in Fig.16a-c are chosen as some examples, each of which has three different numbers of sample points. The corresponding resultant MAs are shown in Fig.16d-f and their execution times are tabulated in Table 4. In Model 4b, since the thicknesses in the upper region and the left region are much larger than the other regions, two additional sharp MA regions are then generated as shown in Fig.16e.

Table 4 Execution times for Models 4a-c

Model No.	Number of sample points	Resolution	Time(ms)
4a	4000	47	61
4a	8000	60	266
4a	16000	75	1155
4b	6500	55	208
4b	13000	70	880
4b	26000	88	3437
4c	4000	47	115
4c	8000	60	401
4c	16000	75	1723

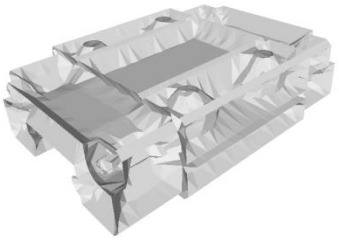


(a) Model 4a



(b) Model 4b

(c) Model 4c



(d) The MA mesh of Model 4a

(e) The MA mesh of Model 4b

(f) The MA mesh of Model 4c

Fig. 16 The MA of Models 4a-c

9. Conclusions

In this work, a parallel method for constructing the MA for a CAD model is proposed. First, the CAD model is discretized into a triangular mesh model. Then, for each sample point of the mesh model, its MA point is generated by using the parallel DNT algorithm. Finally, the MA is generated by connecting the MA points using the connectivities of their corresponding sample points, and the MA faces are extracted from the MA. The main contributions of this work can be summarized as follows:

(1) A DNT algorithm is proposed to generate MA points for all sample points. By using GPU, the first normal tracing of each sample point and the second normal tracing of each triangle are both parallelized. As a result, the computational efficiency is dramatically improved.

(2) Three strategies are proposed to reduce the time complexity of the DNT algorithm. First, the bounding box of a model is divided into grid cells to limit the average number of the pairs of candidate sample point and candidate triangle in each grid cell in parallel computing. Second, the sample points are grouped by their medial radii. In each iteration round, only the sample points whose medial radii are in the current range have their MA points generated, which reduces the average number of intersected grid cells for each sample point in parallel computing. Finally, the adaptive division method is proposed to further optimize the time complexity of the proposed method.

(3) The resultant MA of high quality is generated by using the topological connectivities of the sample points and the mapping between boundary triangles and MA triangles is preserved. Further more, the generated MA can be used to construct the CAD MA faces according to the information of boundary faces.

Our future work will be focused on how to investigate the mutual impacts between the boundary and the MA faces of a CAD model when some changes happen on one or some MA faces.

Acknowledgement

The authors appreciate the support from the National Science Foundation of China (Grant Nos. 61702074, 61572427 and 61370182), the National Key Technology Support Program (Grant No. 2014BAD04B00), the Key Project of Science and Technology of Zhejiang Province (Grant Nos. 2014C01052 and 2014C01017) and the Funamental Research Funds for the Central Universities (Grant Nos. 3132017117).

References

[1] H. Blum. A transformation for extracting new descriptors of shape. In: Models for the Perception of Speech and Visual Form, 1976: 362–380.

[2] X.L. Zhang, Y. Xia, J.Y. Wang, et al. Medial axis tree—an internal supporting structure for 3D printing, Computer Aided Geometric Design,2015, 35–36:149–162

[3] S. Cameron. Collision detection by 4-dimensional intersection testing. IEEE Transactions on Robotics and Automation, 1990, 6(3):291-302.

[4] D. Attali. r-Regular shape reconstruction from unorganized points, Computational Geometry: Theory and Applications 10 (4) (1998) 239–247.

[5] https://en.wikipedia.org/wiki/Medial_axis

[6] H.S. Zhu, Y.S. Liu, J. Bai, et al, Constructive generation of the medial axis for solid models, Computer-Aided Design, 2015, 62:98-111.

[7] H.S. Zhu, Y.S. Liu, J.J Zhao, et al, Calculating the medial axis of a CAD model by multi-CPU based parallel computation, Advances in Engineering Software, 2015, 85:96-107.

[8] H.S. Zhu, Y.S. Liu, J.J Zhao, Generation of hierarchical multi-resolution medial axis for CAD models, Advances in Engineering Software, 2016,94: 20-31.

[9] M. Foskey, M.C. Lin, D. Manocha. Efficient computation of a simplified medial axis. Proceedings of the eighth ACM symposium on Solid modeling and applications, 2003, 96-107.

[10] J. Ma, S. W. Bae, S. Cho. 3D medial axis point approximation using nearest neighbors and the normal field. The Visual Computer, 2012, 28(1): 7-19.

[11] M. Ramanathan, B. Gurumoorthy. Interior medial axis computation of 3D objects bound by free-form surfaces. Comput Aided Des 2010;42(12): 1217–31

[12] D. Attali, J. D. Boissonnat and H. Edelsbrunner. Stability and Computation of the Medial Axes - A State-of-the-Art Report. In Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration, Mathematics and Vizualization. Springer-Verlag, Berlin, Germany, 2009:109-125.

[13] K. Siddiqi, S. M. Pizer. Medial representations: mathematics, algorithms and applications. Springer, Berlin, Germany, 2008.

[14] S. Biasotti, D. Attali, J.D. Boissonnat, et al. Keletal structures, shape analysis and structuring, 145-183, 2008.

[15] G. Elber, E. Cohen, S. Drake. MATHSM: medial axis transform toward high speed machining of pockets. Computer-aided Design 2005, 37(2): 241-250.

[16] I. Hanniel, G. Elber. Computing the Voronoi cells of planes, spheres and cylinders in R2. Computer aided geometric design, 2009, 26(6): 695-710.

[17] J. Ma, S. Choi. Kinematic skeleton extraction from 3D articulated models. Computer-aided Design, 2014, 46(1): 2014: 221-226.

[18] M. Tanase, R. C. Veltkamp. A straight skeleton approximating the medial axis. ESA 2004:809-821.

[19] C. Au. A simple algorithm for medial axis transform computation. Engineering with Computers, 2013, 29:139–149.

[20] Z.C. Chen, Q. Fu. An efficient, accurate approach to medial axis transforms of pockets with closed free-form boundaries. Engineering with Computers, 2014, 30(1):111–123.

[21] T.K. Dey, W. Zhao. Approximate medial axis as a Voronoi subcomplex. Computer-aided design, 2004, 36(2): 195–202.

[22] H.I. Choi, S.W. Choi, H.P. Moon. Mathematical theory of medial axis transform. Pac J Math, 1997, 181(1): 57–88.

[23] J. Lakshmi, Komala; M. Punithavalli. A survey on skeletons in digital image processing. Int. Conf. on digital image processing, Bangkok, Thailand, Mar. 07, 2009.

[24] L. Lam, S.W. Lee, C.Y. SUEN, Thinning Methodologies – A Comprehensive Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14-9(869-885), September 1992.

[25] L. R. Nackman, Curvature relations in three-dimensional symmetric axes. Computer Graphics and Image Processing, 1982, 20: 43-57.

[26] G. L. Scott, S. C. Turner, and A. Zisserman, Using a mixed wave diffusion process to elicit the symmetry set. Image and Vision Computing, 1989, 7(1): 63-70.

[27] K. Siddiqi, S. Bouix, A. Tannenbaum, *et al*. The Hamilton-Jacobi skeleton. In Int. Conf. on Computer Vision (ICCV), 1999: 828-834.

[28] J. Vleugels and M. Overmars, Approximating generalized Voronoi diagrams in any dimension. Technical Report UU-CS-95-14, Dept. of comput. sci., Utrecht Univ. 1995.

[29] G. Borgefors, I. Nyström, G. Sanniti di Baja. Computing Skeletons in Three Dimensions. Pattern Recognition, 1999, 32(7): 1225-1236.

[30] G. Borgefors: On Digital Distance Transforms in Three Dimensions, Computer Vision and Image Understanding, 1996, 64(3):368-376.

[31] G.K.Viswanathan, A. Murugesan. K. Nallaperumal. A parallel thinning algorithm for contour extraction and medial axis transform. 2013 IEEE Int. Conf. on Emerging Trends in Computing, Communication and Nanotechnology, ICE-CCN 2013

[32] S. Stolpner, P. Kry, K. Siddiqi. Medial Spheres for Shape

Approximation. IEEE Transaction on Pattern Analysis and Machine Intelligence. 2012, 34(6):1234-1240.

[33] T.T. Cao, K. Tang, A. Mohamed, et al. Parallel Banding Algorithm to compute exact distance transform with the GPU. ACM SIGGRAPH Symposium on Interactive 3d Graphics and Games. 2010:83-90.

[34] A. C. Jalba, J. Kustra, A. C. Telea. Surface and curve skeletonization of large 3D models on the GPU. IEEE Trans. on pattern analysis and machine intelligence, 2013, 35(6): 1495-1508.

[35] M. Ramanathan and B. Gurumoorthy. Constructing medial axis transform of extruded and revolved 3D objects with free-form boundaries. Computer-Aided Design, 2005, 37(13):1370-1387.

[36] Y.C. Chang, J.H. Kao, J.M. Pinilla, J. Dong; F.B. Prinz, Medial axis transform (MAT) of general 2D shapes and 3D polyhedra for engineering applications. The 6th IFIP working conference on geometric modeling: Fundamentals and Applications, Dec. 7-9, 1998, Tokyo, Japan

[37] E.C. Sherbrooke, N. M. Patrikalakis and E. Brisson, Computation of MA transform of 3-D polyhedral. in ACM Solid Modeling, 1995: 187-199.

[38] O. Aichholzer, W. Aigner, F. Aurenhammer, et al. Medial axis computation for planar free_form shapes. Computer-Aided Design, 2009, 41: 339- 349.

[39] A. Meijster, J. B. T. M. Roerdin , W. H. Hesselink, A general algorithm for computing distance transforms in linear time. In: Mathematical Morphology and its Applications to Image and Signal Processing, Kluwer Acad. Publ., 2000: 331-340.

[40] T. Hirata A unified linear-time algorithm for computing distance maps. Information Processing Letter, 1996, 58(3):129-133.

[41] R. Ramamurthy, T. Farouki Voronoi diagram and medial axis algorithm for planar domains with curved boundaries I: Theoretical foundations. Journal of computational and applied mathematics, 1999, 102: 119-141.

[42] J. Chaussard, M. Couprie, H. Talbot, Robust skeletonization using the discrete λ-medial axis, Pattern Recognition Letters, 2011, 32(9):1384-1394.

[43] B. Miklos, J. Giesen, M Pauly. Discrete scale axis representations for 3D geometry. ACM Transactions on Graphics, 2010, 29:1-10.

[44] F. Sun, Y. K. Choi, Y. Yu, et al. Medial Meshes - A Compact and Accurate Representation of Medial Axis Transform. IEEE Transactions on Visualization and Computer Graphics, 2016, 22(3):1278-1290.

[45] L. Pan, W. Bin, S. Feng, et al. Q-MAT: Computing Medial Axis Transform by Quadratic Error Minimization. ACM Transactions on Graphics 35, 1, Article 8 (December 2015), 16 pages.

[46] L.A. Piegl, T. Wayne. Geometry based triangulation of trimmed NURBS surfaces. Computer-Aided Design, 1998, 30(1):11-18.

[47] https://en.wikipedia.org/wiki/Quartic_function

[48] W. Gao, S.M. Gao SM, Y.S. Liu, et al. Multiresolutional similarity assessment andretrieval of solid models based on DBMS. Computer-Aided Design 2006,38(9): 985–1001.